講義に関しての注意

- 飲食物は持ち込み厳禁！！
- 教科書を必ず持参すること。忘れた場合は図書館に蔵書があるので借りてくること。
- コンピューターを利用して出席確認をするので，講義中は必ず hcs サーバーにログオンしておくこと。
- 動画やゲームは不可、インターネットは用語や技術情報を調べる場合に限り可とする。
- プログラムの修正に関しては，この資料からのコピー&ペーストは不可とする。コピーを利用するとエラーが発生するので，しっかり手作業で入力すること。
- 作業の進捗状況は個人個人で異なる。講義時間以外でも端末を利用して作業ができるので，ペースを考えて自主的に作業を進めておくこと。
- 講師と TA の人が様子を見回るので，分からないところがあったら遠慮なく質問すること。友人に相談するのも可。

レポートの提出方法

宛先　ozawa@mx.ibaraki.ac.jp

件名　授業の曜日時間－学生番号－課題番号　（すべて全角！！）

（例）　火４－０５Ｔ８０４６Ｆ－必修１

木１－０５Ｔ５０２２Ｋ－自由２

添付　MS-Word でレポートを作成し，ファイル名として件名と同じものを使用。

（例）　火４－０５Ｔ８０４６Ｆ－必修１.doc

木１－０５Ｔ５０２２Ｋ－自由２.doc

期日　１２月１日から授業最終日まで。

注意　課題１題につき１つのメールで送付。課題をまとめて送付するのはトラブルの原因。

評価　必修課題を全て提出しなければ単位は与えない。その後は自由課題の提出数と内容に応じて成績を評価する。なお，課題を解く順序は自由とする。

第１回

目的：グラフィックスツール SGL をダウンロードしてグラフィックス環境を備える。

具体的な作業：

① デスクトップにある hcs をクリックし，ASTEC-X を起動してログオンする。
（注：パスワードは入力しても表示はされない）

② 端末エミュレータを起動する。
２本の棒グラフが描かれたアイコンの上にある△マークをクリック。
メニューがポップアップするので，その中の「このホスト」をクリック。

③ 端末エミュレータ中の UNIX プロンプト下で ftp コマンドを実行。

・コマンド：　ftp xbase01.base.ibaraki.ac.jp
　　コンピューター名（ホスト名）
　　または　IP アドレス

・ユーザー名：　anonymous（スペルの間違いに注意）

・パスワード：　E メールのアドレス（例：××@hcs.ibaraki.ac.jp）

これらを入力するとプロンプトが以下のようになる。

・プロンプト　　ftp>

以下の ftp サブコマンドを投入する。

ftp>cd sgl/new	: new ディレクトリの中に移動
ftp>ls	: 中にあるファイル名の表示
ftp>bi	: バイナリモードのファイル転送を指定
ftp>get sglnew.tar.Z	: sglnew.tar.Z のダウンロード
ftp>get read.me	: read.me のダウンロード
ftp>bye	: FTP の終了

ftp プロンプトが UNIX プロンプトに戻る

最後に ls を入力し，sglnew.tar.Z と read.me があることを確認する。

（注）メッセージは全て英語だが，重要な情報が記述されているのでしっかり読むこと。

第２回

目的：SGL を解凍して，サンプルプログラムを実行してみる。

具体的な作業：

① 端末エミュレータを起動し，UNIX プロンプト下で以下のコマンドを実行する。

② more read.me　　：ファイルの中身を表示
（Enter キーで行送り，スペースキーでページ送り）

③ uncompress sglnew.tar.Z　　：解凍（Z は圧縮形式）

④ ls　　：sglnew.tar を確認（Z がとれる）

⑤ tar –xvf sglnew.tar　　：tar 形式のファイルを復元し，複数のファイルにする
（tar 形式：複数のファイルをつなげて１本のファイルにした形式）

⑥ ls　　：２つのディレクトリ（program と sgl）を確認

⑦ cd sgl　　：sgl のディレクトリ中に移動する

⑧ ls　　：その中のファイル名を表示させる

⑨ more makefile　　：makefile というファイル名の中身を見る

⑩ make　　：makefile で指定されているようにコンパイルリンクを行う

⑪ ls　　：実行形式ファイル（example_1，example_2，toda）ができていることを確認。

⑫ ./example_2　　：example_2 を実行する。
（./example_2 の始めのピリオドは，現在いるディレクトリの意味）

⑬ テキストエディタを使って，example_2.c の中身を以下のように変更して上書き保存。
#define 文で定義されている粒子数を十倍・時間を千倍にする。

⑭ ⑩～⑫を実行。
（注：ウィンドウを閉じるときは，ウィンドウの左上をダブルクリックする）

テキストエディタの使い方

- 紙とペンが描かれたアイコンの上にある△マークをクリック。
- メニューがポップアップするので，その中の「テキストエディタ」をクリック。
- 「ファイル」→「開く」でファイル選択用ウィンドウが開かれる。
- フォルダの中から「sgl」をダブルクリック。
- ファイルの中から「example_2.c」をダブルクリック。
- 編集した後で，「ファイル」→「保存（必要）」をクリック。

第３回

目的：テキストエディタを使って，グラフィックスプログラムのソースコードを作成する。それをコンパイルして実行形式のプログラムを作成する方法を実習する。

具体的な作業：

① テキストエディタを起動して，教科書ｐ16～17 のソースコードを入力し，sgl ディレクトリの中に hard.c というファイル名で保存する。
（注：コードは左寄せにせず，ちゃんと教科書の通りにインデント（左端からの段分け：半角スペース２つずつ）をそろえること。また，コメント（/*…*/）もしっかり入力すること。）

② sgl のディレクトリ中の makefile をテキストエディタで開く。その中の用語 toda を次の手順で hard に置換し，sgl のディレクトリ中に makefile_hard というファイル名で別名保存する。
・「編集」→「検索／変更」をクリック
・メニューが開くので，検索に「toda」，変更後の単語に「hard」を入力
・「すべてを変更」をクリックしてから，「閉じる」をクリック

③ 端末エミュレータを開き，sgl のディレクトリ中で以下のコマンドを実行する。
・make –f makefile_hard　　：makefile_hard に従ってコンパイルリンクを行う
（注：「Error」が表示されたら hard.c を編集してやり直す。どのファイルの何行目にどのようなエラーがあるのか英語で表示されるので，しっかり読むこと。）
・ls　　：オブジェクト形式ファイル（hard.o）と実行形式ファイル（hard）ができて
いることを確認
・./hard　　：プログラムの実行

C言語の文法（読み方・書き方）

（１）/*と*/で挟まれた部分はコメント文（注釈文）。
（２）#include はファイルの取り込み（取り込む対象は○○．h なるヘッダーファイル）。
（３）#define　A　B　は，A をＢと見なして実行する。主に数値の設定に利用する。
（４）void main(int argc, char *argv[ ]) {　……
} はメイン関数。{　……　}の部分にその作用を記述する。int argc, char *argv[ ] の意味は，さし当たって考えなくても良い。
（５）float　x, ….;　　x,….の変数は，実数型であることを意味する文。文の終わりには必ず；（セミコロン）がつくことを覚える。
（６）int　i;　　i の変数は，整数型であることを意味する文。
（７）sgl_winsize(500,500);　　一般に sgl_○○は SGL（画像ソフト）の関数を意味する。この場合は，画像出力用のウィンドウのサイズを 500x500（ピクセル単位）にする

という意味。

（８）sgl_init( );　　画像出力用のウィンドウを初期化して生成する。

（９）= は数学と意味が違う。右側の計算結果を左側の変数に代入するという意味。

（10）for (t=0; t<tmax; t+=dt){ ……

}は，t+=dt は t=t+dt と同等。(t=0; t<tmax; t+=dt)は，t の初期値を 0 とし，それを dt ずつ足し合わせて，tmax になるまで{ ……

}の作業を繰り返して行うという意味。

（11）if(x<=0. || x>=1.) はもし x が 0 以下か 1 以上の場合はという意味。その場合はこの文の次の文を実行し，そうでない場合にはそれをスキップする。

エラーの処理

- ③のコマンドを実行したときに「Error」と表示されたら，hard.c を編集してやり直す。「warning」は「できればこの方が良い」という警告なので，無視してもよい。
- エラー内容は主に「ファイル名: 行番号: エラー内容」の形式で表示される。表示は英語だが簡単に記述されているのでしっかり読むこと。
- 複数のエラーが表示された場合は，一番上のものから対処する。一箇所を修正しただけで複数のエラーが同時に消えることもある。
- エラー箇所を探す際には，テキストエディタで「オプション」→「ステータス行」をクリックしておくと良い。参照している行番号を調べたり，指定した行にジャンプしたりできるようになる。
- ③のコマンドをやり直す前に，hard.c を上書きしておく（テキストエディタで「ファイル」→「保存（必要）」をクリックする）。makefile_hard はそのままで良い。
- 同じコマンドを入力する場合は，端末エミュレータで!（エクスクラメーション）の後に行番号を入力すれば良い。一つ前のコマンドの場合には!!で実行することもできる。

第４回　（レポート：必修課題１）

目的：アルゴリズムによる誤差（近似による誤差）について，具体例で理解する。

具体的な作業：

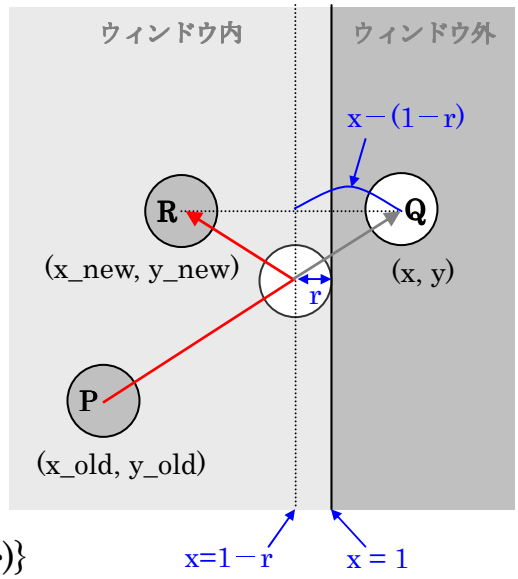
① プログラム hard.c では，衝突の計算に誤差がある。まずは種々の設定値（#define の数値）を変化させながらプログラムを実行し，ディスクが端に食い込むこと，その程度が衝突ごとに異なること（運動可能面積が不確定）を観察する。計算時間（t_max）と半径（r）を大きく，初速度（vx と vy）を小さくすると観察しやすい。
（ウィンドウの幅が 1 なので r は 0.5 以下にすること。また，vx：vy の比率は 2：1 を維持すること。）

② 誤差の生じ方が理解できたら，次に衝突のアルゴリズムの改良方法を考える。例として，右端（x 軸の正方向）との衝突の模式図とそこから得られる位置の補正式を示す。

右端との衝突

移動前の位置：P(x_old, y_old)
移動後の位置：Q(x, y)
衝突後の位置：R(x_new, y_new)

（**R**はx＝1－rに対する**Q**の鏡像である）



R の x 座標は x_new = x－2{x－(1－r)}
整理すると x_new = 2(1－r)－x

③ 同様に左端（x 軸の負方向）との衝突，上端（y 軸の正方向）との衝突，下端（y 軸の負方向）との衝突のアルゴリズムを考える。

④ プログラム的には，衝突の計算をする部分（2 つの if 文）を以下のように修正する。空欄は②と③から得られる数式を適当な形に直して埋めること。

（注：数学的な簡略表現は使えない。2x→2＊x のように加減乗除をしっかり記述すること。）

```
if(x<=0.0＋r) {
   x= [          ] ;
   dx= - dx;
}
```

```
if(x>=1.0 - r){
   x= [          ] ;
   dx= - dx;
}
if(y<=0.0+r) {
   y= [          ] ;
   dy= - dy;
}
if(y>=1.0 - r){
   y= [          ] ;
   dy= - dy;
}
```

⑤　改良された衝突のアルゴリズムの説明と④での修正内容をまとめて，レポートする。
図表の挿入も可。

第５回　（レポート：必修課題２）

目的：常微分方程式の数値解法の基本であるオイラー法について学習する。
その例題を実行し，「打ち切り誤差」，「積み重なり誤差」を体験する。

具体的な作業；

① 教科書ｐ.20～22 を読み，Fig.1 を参考にしてオイラー法の計算原理を理解する。
（注：ソースコードを入力する必要はない）

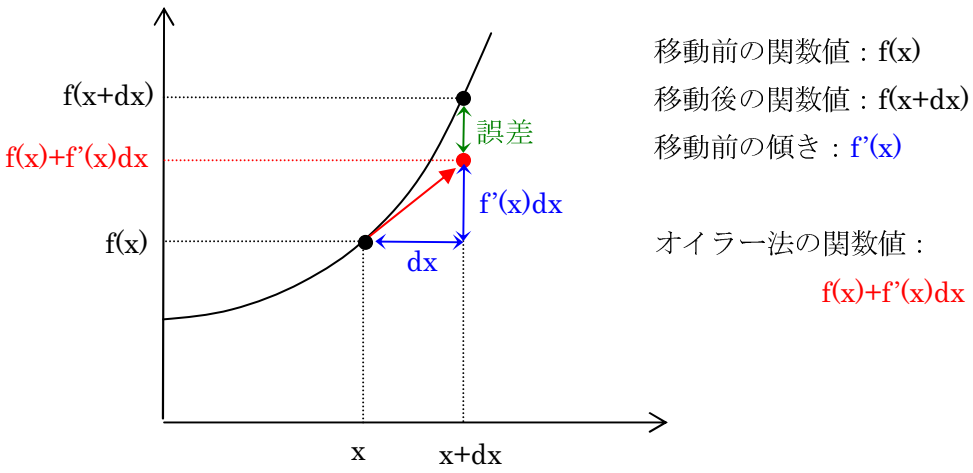


Fig.1　オイラー法の計算原理

② オイラー法の計算原理が理解できたら，以下のコマンドを入力して program のディレクトリにある euler.c を sgl のディレクトリにコピーする。
（sgl のディレクトリ中で cp ../program/euler.c euler.c）

③ sgl のディレクトリ中の makefile をテキストエディタで開き，その中の文字列 toda をすべて euler に置換し，別名保存（makefile_euler）する。

④ sgl のディレクトリ中で，以下のコマンドを実行する。
・make –f makefile_euler
・./euler

⑤ 振り子の振動回数を 5 回に設定する。計算時間は時間刻み dt×計算回数 maxmds であり，振動周期は 2πである。よって euler.c 中の maxmds を 2π×5 / dt に設定すれば良い。（π＝3.1415927 とする。dt はそのまま入力して良い）

⑥ euler.c 中の dt の値を 1 桁ずつ小さくしながらプログラムを実行し，計算終了時における全エネルギー e の誤差|e－0.5|がどのように変化するかを調べる。結果は MS-Excel のファイルに保存しておく。誤差に関しては参考 1 を，プログラムの実行に際しては参考 2 を熟読しておくこと。

⑦ ⑥の結果から dt（対数）と誤差の片対数グラフを作成し，考察を加えてレポートする。考察に関しては参考 3 と参考 4 を熟読しておくこと。

参考１　：　プログラムの内容

調和振動子の運動を数値的に計算するプログラムである。この問題は解析的にも解ける。（微分方程式の解も分かっているが，それをあえて数値的に計算している。）

微分方程式は $m\frac{d^2x}{dt^2} = -kx$ であり，$m = 1$，$k = 1$ である。初期条件は $x(0) = 1$，$v(0) = 0$ であるから，解は $x(t) = \cos t$ である。その周期は 2π である。$v(t) = \sin t$ であるから，エネルギーは $E = \frac{m}{2}v^2 + \frac{k}{2}x^2 = \frac{m}{2}v(0)^2 + \frac{k}{2}x(0)^2 = 0.5$ となり，常に一定である。

参考２　：　計算上の注意

この問題では振り子が 5 回振動した時点で，全エネルギー（運動エネルギー＋位置エネルギー）にどれだけの誤差が入るのかを調べるのが課題になっている。それが，時間きざみ dt にどのように依存するかを調べる。具体的には，dt の値が小さすぎても大きすぎても誤差が発生する様子を見る。

dt の値を小さくすると maxmds の値が大きくなり（dt×maxmds = 2π×5 であるから），計算にかかる時間が大きくなる。その場合，振り子の運動は極めて遅くなり，事実上動かないように見える。グラフィックスは必要がないので，グラフィックス出力をしないようにプログラムを修正すること。具体的には，sgl_ ではじまる関数 4 つをコメントアウトすればよい（sgl_ ではじまる 4 つの関数をそれぞれ/*と*/ではさむ）。

それでも時間がかかるようになったら，ループ中の中間出力の回数を減らすことで更に高速に計算することができる。先ずは，出力文 printf(...)を for(...){...}の後にコピーする。次に，コピー元（for ループの中）の出力文の前に if(mds%10000000＝＝0)を追加する。コピー先（for ループの後）の出力文には修正を加えないこと。%は割り算の余りを計算する演算子であり，＝＝は両辺が等しい（数学での＝と同じ）条件を表す演算子である。よって，修正内容は mds ステップが 10000000 の倍数（10000000，20000000，...）のときにのみ出力文 printf(...)を行う，という意味となる。また，事前に出力文をコピーしたのは最終結果を出力するためである。

参考３　：　「打ち切り誤差」（丸め誤差）とは

計算機の中では数値が有限の桁数の 2 進数で表されるため，無限に正確な値を表現することはできない。float 文で宣言された数値は，10 進数換算で 7.5 桁の有効数字で表される。小さい数間の演算が行われると，演算結果の 7 桁以下の微小な値は無視される（打ち切られる）ことになるので，演算が不正確になる。これを回避するには，表現できる桁数の大きい変数を使う・数値の単位を小さくするといった方法がある。

参考 4　：　「積み重なり誤差」とは

for ループの中での演算は多くの回数繰り返して行われる。1 回の計算で生じる誤差は充分に小さくても，繰り返し計算によってそれが積み重なり，大きく成長する場合がある。これを積み重なり誤差という。誤差が積み重なってくると誤差の入り方も大きくなる場合があり，これを誤差の爆発という。そのため，誤差が爆発しない程度に積み重なり誤差を押さえる必要がある。差分法においては，差分の値を小さくすれば「積み重なり誤差」は小さくなる。

MS-Excel の使い方

- 微小な値を入力すると「3E－8」のように表示されることがあるが，これは 3×10^{-8} という意味である。小数で表示したい場合は，右クリックから「セルの書式設定」→「表示形式」→「数値」で小数の桁数を設定してやれば良い。
- 計算式は＝の後に入力し，最後に Enter キーを押す。計算対象のセルは左クリックで指定する（直接入力することもできる）。
  （例）セル A2 が 1.5・セル B3 が 0.3 のとき，「＝A2－B3」は 1.2 となる
- 絶対値は ABS(…)という関数で計算できる。関数は計算式の一部として用いる。
  （例）セル A3 が－3.4 のとき，「＝ABS(A3)」は 3.4 となる
- 隣接する複数のセルに対して同じ計算をする場合は，計算式を入力したセルを選択した状態でセルの右下に表示される■をダブルクリックすれば良い。ただし，計算式を入力したセルも隣接している必要がある。
  （例）セル C1 が「＝A1－B1」のとき，右下の■をダブルクリックするとセル C2 に「＝A2－B2」・セル C3 に「＝A3－B3」が入力される
- 行や列をまとめて選択する場合は，行の見出し（1,2,…）や列の見出し（A,B,…）をクリックすれば良い。複数の行や列を選択する場合は，Ctrl キーを押しながらクリックすれば良い。
- 座標を指定してプロットするグラフには「散布図」を用いる。横軸の数値と縦軸の数値を選択し，「挿入」→「グラフ」→「散布図」をクリックすれば良い。プロット間に線を入れる場合は数値の並び順になるので，順番を整えておくこと。
  （注：「折れ線」は等間隔に並ぶデータ（年次推移など）に用いられる。散布図と混同しないように注意。）
- 軸上の数値をダブルクリックすると「軸の書式設定」が開かれる。「目盛」内にある「最小値」と「最大値」を変更することにより，グラフ内の注目するべき部分を拡大表示することができる。また，「対数目盛を表示する」をクリックすることで対数グラフを作成することができる。

第６回　（レポート：必修課題３）

目的：オイラー法に代わってルンゲ・クッタ法を使って前回と同様のデータを取得し，
打ち切り誤差・積み重なり誤差を比較する。（誤差がいかに改善されるかを調べる。）

具体的な作業：

①　教科書ｐ.22～24 及び Fig.2 を参考にルンゲ・クッタ法の計算原理を理解する。

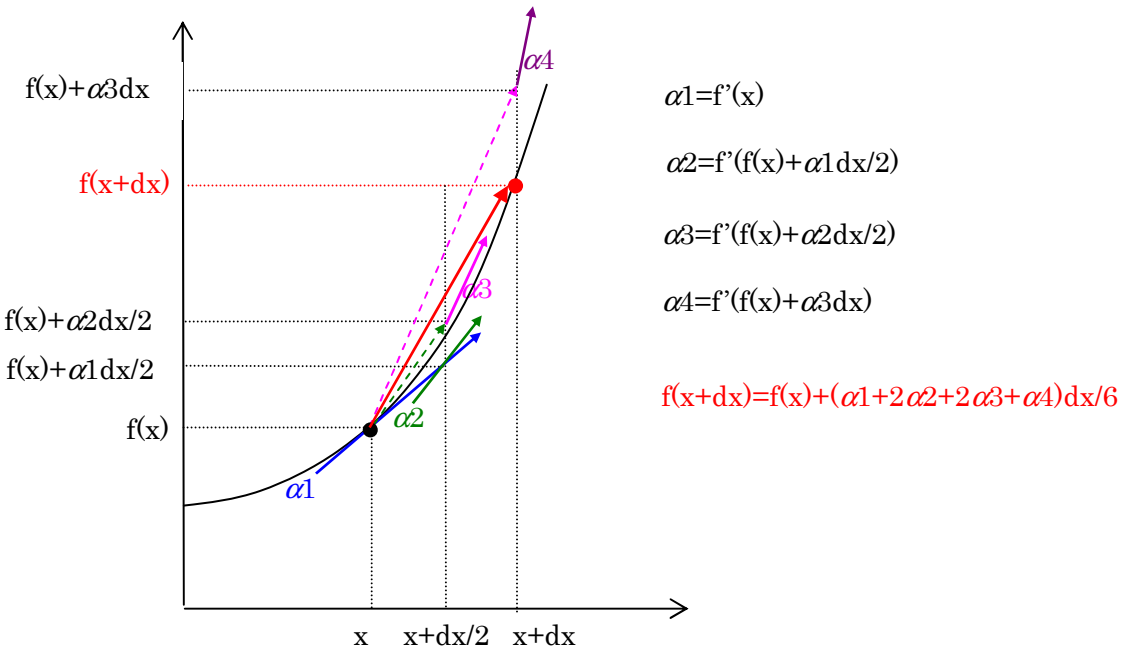


Fig. 2　ルンゲ・クッタ法の計算原理

②　ルンゲ・クッタ法の計算原理が理解できたら，program ディレクトリ中の runge.c を sgl ディレクトリにコピーする。

③　このプログラムのコンパイル用の makefile を作成し，makefile_runge と別名保存する。（sgl ディレクトリ中の makefike をエディターで開き，toda を runge と置換）

④　UNIX プロンプト下で（sgl のディレクトリの中で）
make –f makefile_runge

⑤　前回と同様に maxmds を変更し，dt を 1 桁ずつ小さくしながら 5 回振動した時点での e の誤差を測定する（e の真値は 0.5 である。maxmds は 2π×5 / dt とする。）。また，その結果を MS-Excel のファイルに保存し，グラフを作成する。

⑥　上のデータとオイラー法のデータを比較し，考察を加えてレポートする。

第７回　（レポート：自由選択課題１）

目的：誤差の評価の仕方を改善する。（必修課題２及び３では，単振動の計算機シミュレーションにおいて，5 回振動した時点での力学的エネルギー e の誤差|e－0.5|を計測したが，誤差は時間的にも変動している。そこで，e の誤差を時間的に積分した量で誤差の評価ができるようにプログラムを改良し，そのデータを取り，以前のデータと比較する。）

具体的な作業

① 調和振動子が 4 回振動した時点（mds＝2π×4 / dt）から 5 回振動する時点（mds＝2π×5 / dt）までの最後の 1 周期の各 mds ステップで，(e－0.5)²dt の値を計算し，それらをこの期間で積分するようにプログラム（euler.c 及び runge.c）を変更する。
すなわち，err＝err＋(e－0.5)＊(e－0.5)＊dt;を各プログラムにおける e の計算式の後に挿入する（変数 err をあらかじめ float 文で宣言し，0.0 に初期化しておくこと）。また，この計算式の直前に if 文を追加し，この計算が所定の期間のみで行われるようにする（5 回振動した時点で計算が終わることに注意）。そして，for(…){…}の後に printf 文を追加し，err の値を出力する。

② 上のプログラムを適当なファイル名で保存し，それをコンパイルする makefile を作り，make を実行する。

③ 以前と同様に，dt を変えながら err の値を計測する。

④ 得られたデータを必修課題２及び３と比較し，それから導かれる結論をレポートする。

第８回　（レポート：自由選択課題２，３）

自由選択課題２

目的：全エネルギー e の計算が正確でも，それ以外の量が正確に計算できているという保障はない。そのほかの物理量で，誤差の評価を行う。

具体的な作業：

① $x = \cos t$ が解であるから，$\{x - \cos(\mathrm{mds} \times \mathrm{dt})\}^2 \times \mathrm{dt}$ を最後の 1 周期で積分する。

```
if(mds >= 4*2*3.141592/dt)
    err = err + (x - cos(mds*dt))*(x - cos(mds*dt))*dt;
```

② 自由選択課題 1 と同様に err の dt 依存性を調べ，自由選択課題 1 の結果と比較する。比較した結果に考察を加えてレポートする。

自由選択課題３

目的：解が知られていないシステムにおける誤差の評価法を学習する。

具体的な作業：

① 教科書 p .39 のプログラム 2.5 を実行できるようにする。
program ディレクトリ中の anharmo.c を sgl ディレクトリの中にコピーし，それに対応する makefile をつくり，make を実行する。

② 全エネルギー e の値は一定であるべきである。dt の値を変えて計算を行い（それに応じて maxmds の値を変えること。すなわち，dt×maxmds＝一定（任意の定数）とすること。），全エネルギー e の値を計測する。全エネルギー e の値を dt の関数として，グラフにする。これより，計算が正しく行われる dt の値の範囲を評価し，考察を加えてレポートする。

（注意）program ディレクトリ中の anharmo.c には，エラー（バグ）がある。
#define 宣言の最後に

```
#define  factinit  1.0
```

を追加して，解決すること。

第９回　（レポート：自由選択課題４）

目的：必修課題１を発展させ，２つの粒子の衝突を実現させる。

２つの粒子の衝突は完全弾性衝突とする。

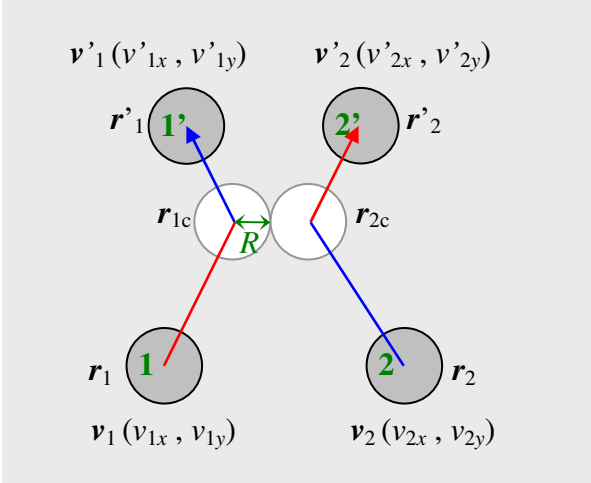
衝突の前後で全運動量が保存し，衝突の反発係数が１であることから，

$m\,\boldsymbol{v}'_1 + m\,\boldsymbol{v}'_2 = m\,\boldsymbol{v}_1 + m\,\boldsymbol{v}_2$ …(1)

$\boldsymbol{v}'_1 - \boldsymbol{v}'_2 = -1(\boldsymbol{v}_1 - \boldsymbol{v}_2)$ …(2)

が成立する。

具体的な作業：



① $m = 1$ とし，式(1)(2)より，

$$\left.\begin{aligned} v'_{1x} &= v_{2x} \\ v'_{1y} &= v_{2y} \\ v'_{2x} &= v_{1x} \\ v'_{2y} &= v_{2x} \end{aligned}\right\} \cdots(3)$$

② 衝突時刻 t を導く。

衝突位置を $\boldsymbol{r}_{1c}$，$\boldsymbol{r}_{2c}$ とすると，

$$\left.\begin{aligned} \boldsymbol{r}_{1c} &= \boldsymbol{r}_1 + \boldsymbol{v}_1 t \\ \boldsymbol{r}_{2c} &= \boldsymbol{r}_2 + \boldsymbol{v}_2 t \end{aligned}\right\} \cdots(4)$$

$|\boldsymbol{r}_{1c} - \boldsymbol{r}_{2c}| = 2R$ …(5)

式(4)を(5)に代入して，

$t = ?$ …(6)

③ 衝突後の位置を導く。

$$\left.\begin{aligned} \boldsymbol{r}'_1 &= \boldsymbol{r}_{1c} + \boldsymbol{v}'_1(dt - t) \\ \boldsymbol{r}'_2 &= \boldsymbol{r}_{2c} + \boldsymbol{v}'_2(dt - t) \end{aligned}\right\} \cdots(7)$$

④ 式(3)(4)(6)(7)を使ってプログラム hard.c を発展させ，二つの粒子の衝突を実現する。発展させたプログラムの内容に解説を加えてレポートする。

ヒント：必修課題１のプログラムにおいて，先ずは粒子を２つにする。その後，境界との衝突を実現する if 文の後に粒子間の衝突を実現する if 文を追加する。if 文の条件式は，時間刻み dt と衝突時刻 t を用いて設定する。

第 10 回　（レポート：自由選択課題５）

目的：必修課題２及び３では，float 型の変数を用いて計算を行った。しかし，float 型の変数では 7.5 桁までの数しか保存できないため, 打ち切り誤差が発生する。そこで, 変数を float 型からその倍の桁数まで保存できる double 型にして計算し，誤差の改善を確認する。

具体的な作業

① プログラム（euler.c 及び runge.c）の変数宣言（main の下，int ...;および float ...;）で，float 宣言を double 宣言に修正する。

② 上のプログラムを適当なファイル名で保存し, それをコンパイルする makefile を作り, make を実行する。

③ 必修課題２及び３の結果を見直し，打ち切り誤差（dt が小さすぎるときに発生する誤差）が確認できる dt の値を確認する。

④ ③の dt の値を用いてプログラムを実行し（maxmds は 2π×5 / dt とする），全エネルギー e の値の誤差 | e－0.5 | を計測する。

④ 得られたデータを必修課題２及び３と比較し，考察を加えてレポートする。

第 11 回　（レポート：自由選択課題６）

目的：調和振動子の５回振動を計算するのにかかる時間を調べ，計算効率を評価する。
（オイラー法からルンゲ・クッタ法にすることで，計算効率がどれくらい向上するかを調べる。）

具体的な作業

① 計算にかかる時間を調べるコードをプログラム（euler.c 及び runge.c）に追加する。
まずは，#include ＜…＞宣言の最後に以下の文を追加する。

```
#include ＜time.h＞
```

これによって時間を調べる機能が使えるようになる。
次に，変数宣言（main の下，int …;および float …;）の最後に以下の文を追加する。

```
float begin＝clock( );
```

clock( )は時刻を取得する関数である。よって上の文では，計算開始時の時刻を変数 begin に保存している。
最後に，for(…){…}の次に以下の文を追加する。

```
printf(“time＝%f sec¥n”, (clock( )－begin) / CLOCKS_PER_SEC);
```

カンマまではこれまでと同様の形式である。その次の括弧では，計算終了時の時刻から計算開始時の時刻を引くことで，計算にかかった時間を求めている。時間は特殊な単位で表現されているので，CLOCKS_PER_SEC 定数で割ることで単位を秒になおしている。

② 計算が一瞬で終わってしまうと時間を計ることができないので，振り子の振動回数を 10000 回に増やす。（maxmds を 2π×10000 / dt とする）

③ 上のプログラムを適当なファイル名で保存し，それをコンパイルする makefile を作り，make を実行する。

④ 必修課題２及び３の結果を見直し，オイラー法とルンゲ・クッタ法それぞれの，誤差が 0.001 以下となる dt の最大値を確認する。

⑤ ④の dt の値を用いてそれぞれのプログラムを実行し，time の値を計測する。この値が小さいほど，計算効率が良いといえる。

⑥ 得られたデータより，オイラー法からルンゲ・クッタ法に変更したときの計算効率の向上比率を求め，考察を加えてレポートする。